AFRL-RI-RS-TR-2013-203

EVALUATION OF IMAGE SEGMENTATION AND OBJECT RECOGNITION ALGORITHMS FOR IMAGE PARSING

SEPTEMBER 2013

FINAL TECHNICAL REPORT

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

FOR THE DIRECTOR:

/ S /

AMANDA LANNIE
Work Unit Manager

/ S /

MICHAEL J. WESSING
Deputy Chief, Information Intelligence
Systems and Analysis Division
Information Directorate

REPORT DOCUMENTATION PAGE	*Form Approved* **OMB No. 0704-0188**

1. REPORT DATE *(DD-MM-YYYY)* SEPTEMBER 2013	**2. REPORT TYPE** FINAL TECHNICAL REPORT	**3. DATES COVERED** *(From - To)* MAY 2012 – JAN 2013

4. TITLE AND SUBTITLE EVALUATION OF IMAGE SEGMENTATION AND OBJECT RECOGNITION ALGORITHMS FOR IMAGE PARSING	**5a. CONTRACT NUMBER** IN-HOUSE
	5b. GRANT NUMBER AFOSR MINI GRANT
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Amanda Lannie	**5d. PROJECT NUMBER** E1RA
	5e. TASK NUMBER AL
	5f. WORK UNIT NUMBER SA
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Research Laboratory/RIED 525 Brooks Road Rome NY 13441-4505	**8. PERFORMING ORGANIZATION REPORT NUMBER** N/A
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory/RIED 525 Brooks Road Rome NY 13441-4505	**10. SPONSOR/MONITOR'S ACRONYM(S)** AFRL/RI
	11. SPONSORING/MONITORING AGENCY REPORT NUMBER AFRL-RI-RS-TR-2013-203

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited. PA# 88ABW-2013-2741
Date Cleared: 10 Jun 2013

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The goal of this effort is to implement several algorithms for image segmentation and object recognition, unify the algorithms, and determine which approach works the best based on certain measures. Based on the number of segments produced by the segmentation implementations, there is over-segmentation or incorrect segmentation (according to a human's perception). The performance of the segmentation could have influenced the results of the object recognition. The results for precision, recall, and F-measure indicate that the best approach to use for image segmentation is Sobel edge detection and to use Canny or Sobel for object recognition. The process for this report would not work for a warfighter or analyst. It has poor performance. Additionally, its lack of variety among the algorithms reduces the chance of correctly labeling the objects in an image.

15. SUBJECT TERMS
Segmentation, object recognition, computer vision

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** **AMANDA LANNIE**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			**19b. TELEPONE NUMBER (*Include area code)***
U	U	U	UU	33	**N/A**

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. SUMMARY

The military collects massive amounts of imagery and video every day. The amount of data makes it impossible for a warfighter/analyst to review all of it. Image parsing algorithms alleviate this burden on warfighters and analysts by reducing the manual labor of labeling each image. Image parsing is the splitting of an image into parts and labeling them. Image segmentation and object recognition make image parsing possible. Image segmentation is the partitioning of the image into parts of significance. The specificity of the portioning depends on the approach. For instance, one approach might segment out a car in an image while another approach may see wheels, windows, car body, etc. Object recognition is the identification of the objects in the image where an object is anything that falls into the category noun (person, place, or thing). The vocabulary to describe the objects is usually known a priori and associated with a set of features like color, shape, or edges. To determine the similarity of the features one can use methods like classification or matching. Currently, researchers do not explain their reasoning for choosing their image segmentation and object recognition methods or their focus is only one area of the parsing process. The goal of this effort is to implement several algorithms for image segmentation and object recognition, unify the algorithms, and determine which approach works the best based on certain measures. Based on the number of segments produced by the segmentation implementations, there is over-segmentation or incorrect segmentation (according to a human's perception). The performance of the segmentation could have influenced the results of the object recognition. The results for precision, recall, and F-measure indicate that the best approach to use for image segmentation is Sobel edge detection and to use Canny or Sobel for object recognition. The process for this report would not work for a warfighter or analyst. It has poor performance. Additionally, its lack of variety among the algorithms reduces the chance of correctly labeling the objects in an image. Nevertheless, perhaps with some more experimentation like implementing other methods, the image segmentation and object recognition of image parsing itself could be useful.

2. INTRODUCTION

Image segmentation is the sectioning of an image into meaningful parts; the amount of segmentation on an image depends on the problem. It is also one field in the subject of computer vision. Thus, it has its own set of methods of implementation and issues. Not only are there several categories for segmenting (see Table 1), there are many implementations for each category. For example, the concept of edge detection has within it Prewitt mask ([1], [2]), Sobel mask ([2], [3]), Marr-Hildreth edge detector ([2], [4]), Canny edge detector ([2], [5]), and those are only some of the possibilities. Image segmentation is nontrivial. A computer has no concept of when to stop dividing an image because it has no idea the specificity (very generic to very detailed) of the segmentation the user has in mind. The computer has to go by the algorithm and parameters given in the code to know when to stop segmenting an image. Textures can complicate the process; for instance, if the approach is edge detection on an image of a couch with a print for the upholstery, then a computer would likely not identify the couch.

Table 1. Categories for Image Segmentation [6]	
Thresholding	Partial differential equation-based
Clustering	Graph partitioning
Compression-based	Watershed transformation
Histogram-based	Model based
Edge detection	Multi-scale
Region growing	Semi-automatic
Split and Merge	

Additionally, no implementation is perfect. Edge, line, and point detection can miss edges, lines, and points, respectively. These are only some of the issues when dealing with segmentation.

Object recognition is the identifying of objects (person, place, or thing) in an image and is no easy feat. Despite the abundance of approaches, none are flawless. Most require some manual labor because there is a need for a priori knowledge to give objects their labels. Some additional challenges to the process are occlusion, lighting, and views, see Table 2. To see images of these challenges see [7]. The more objects that are in the picture, the more objects are overlapping and less of the whole object is available therefore making approaches that use matching have lesser performance. Lighting and the view make object recognition difficult because they could be different from that of the contents of the database. Thus, matching an object to one in a database becomes difficult whether taking a matching approach (giving a label to the segment by basing it on the skeleton/structure of the image) or a statistical one.

Table 2. Examples of Challenges in Computer Vision

Challenge	Description of Image
Original image	The top of a car.
Shifting, scaling, and rotation	Picture of the top of the car taken further away. Additionally, the car has been moved to the right and put at a 45° angle.
Lighting change	A bright spot on the top of the car caused by a street light.
Viewpoint change	Side view of the car.
Partial occlusion	A tree hanging over par of the top of the car and thus blocking part of it.
Clutter	The top of the car surrounded by trees and other vehicles.

Analysts are already overwhelmed with images and video data to the point that processing it all is not possible. Therefore, it would be best to invest in observing image segmentation and object recognition of image parsing, crucial parts of any assisting application, and consider only methods that are automatic. By evaluating image parsing's object recognition and image segmentation and discovering the weaknesses, it would indicate areas in need of improvement.

2.1. Sobel Edge Detection

Sobel is a well-known method of edge detection and achieving the result is relatively easy to understand. A black and white image has convolution done on it in the x and y directions using kernels $Gx = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$ and $Gy = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$. The operator to use in the x direction is G_x and G_y for the y direction. The twos in the operators provide some image smoothing. $G = \sqrt{Gx^2 + Gy^2}$ summarizes the next steps. Using the results of the convolutions, for each pixel, the values are squared and summed. The sum's square root is calculated; this is the gradient magnitude for the pixel. Pixels are part of an edge if their value is non-zero. The strongest response to this process is in the vertical and horizontal directions; therefore, vertical and horizontal edges would be most prominent.

The result of using Sobel is an image with bounded regions. Sobel edge detection is used as a method of image segmentation since each of the bounded regions is a segment.

```
//C:\OpenCV231\samples\cpp\tutorial_code\ImgTrans\Sobel_Demo.cpp

int scale = 1;//scale factor for the computed derivative values; by default no
scaling is applied
int delta = 0;//value that is added to the results prior to storing them in dst
int ddepth = CV_8U;//output image depth, type of the image
Mat grad, gradX, gradY, absGradX, absGradY;
//Sobel parameters:  src, dst, ddepth, xorder, yorder, ksize, scale (optional),
delta (optional), borderType
//src is the original image in black and white
//dst is the destination for the results
//ksize is the size of the extened Sobel kernel (1,3,5,7)
//xorder & yorder is derivative of respective direction
//borderType is pixel extrapolation method
Sobel(src, gradX, ddepth, 1, 0, 3, scale, delta, BORDER_DEFAULT);
convertScaleAbs(gradX, absGradX);//scales, computes absolute values, and con-
verts result to 8-bit
Sobel(src, gradY, ddepth, 0, 1, 3, scale, delta, BORDER_DEFAULT);
convertScaleAbs(gradY, absGradY);
addWeighted(absGradX, 0.5, absGradY, 0.5, 0, grad);
```

2.2. Canny Edge Detection

Canny is a common method of edge detection. From a programmer's standpoint, the process can be described in four steps: apply Gaussian blur, find gradient magnitudes, perform non-maximum suppression, and hysteresis. Applying the Gaussian filter to the entire image results in a blurred version of the original. This step reduces noise. The second step of finding the gradient magnitudes is applying Sobel edge detection to the blurred image. An alternative to Sobel to compute the gradient is the Prewitt mask (another gradient operator). Finding the edges using the gradient makes the lines thicker around the local maxima. Non-maxima suppression suppresses pixels that are not the maximum magnitudes from the previous step to reduce the thickness by using gradient. The final step is hysteresis, which comes from the area of physics. [8] defines hysteresis as increasing the total magnetic field to a maximum when atoms are aligned though the total field lags behind the magnetizing field, and when the intensity of the magnetizing field is zero, some of the field remains. Canny uses this idea; hysteresis uses two thresholds; this will help avoid the appearance of a broken line. Anything above the high threshold marks the pixel as a strong edge. Anything between the two thresholds is a weak edge. Finally, nothing below the low threshold is an edge. Hysteresis looks at pixels connected to strong edges and added to the actual edge if they are a strong edge or weak edge. The additions continue until the value of the pixel is less than the low threshold.

The product of applying Canny edge detection to an image is another image with bordered sections. These sections are the segments; hence, it is a method of image segmentation.

```
//C:\OpenCV231\samples\cpp\tutorial_code\ImgTrans\CannyDetector_Demo.cpp

double threshold1 = 75;//first threshold for hysteresis
double threshold2 = 150;//second threshold for hysteresis
Mat out;//destination for the result
//3 is the size of the Sobel operator, true is use the actual distance formula
(false is some form of distance formula but calculation is faster)
Canny(src, out, threshold1, threshold2, 3, true);
```

2.3. Mean Shift Filtering

Mean shift filtering smoothes the image and thereby blurring it. This removes noise in the image. Mean shift filtering preserves discontinuity; thus, the smoothing near edges is reduced [9]. Blurring the image creates a better image for segmentation in that there would be fewer segments.

The function and approach we applied is pyrMeanShiftFilter. The function uses the filtering step of the mean shift segmentation algorithm. For every pixel, it is the center of some sized window, this is a parameter given to the function. (The function takes a radius for the spatial component and for the color.) Within that window, the average spatial values and color values are calculated. The window shifts to the new spatial coordinates where the new values become the center. The computation of the spatial and color values is continuous until convergence. After the last iteration, the initial pixel becomes the last iteration's color component average. The result of this process is a polarized version of the original image.

```
//C:\OpenCV231\samples\cpp\meanshift_segmentation.cpp

int spatialRad = 10;//spatial window radius
int colorRad = 10;//color window radius
int maxPyrLevel = 1;//max level of pyramid for the segmentation
Mat out; //destination for the result
pyrMeanShiftFiltering(src, out, spatialRad, colorRad, maxPyrLevel);//does mean
shift filtering
```

2.4. Scale Invariant Feature Transform (SIFT)

David Lowe created SIFT in 1999 at the University of British Columbia. He had it patented in 2004. SIFT looks at local features, which helps when doing object recognition on images that are cluttered or have partial occlusion.

Scales and octaves construct a scale space. A scale is images of the same size but different (incremental) blurriness and an octave is images of different sizes (size is halved each time). Therefore, a scale space is a scale for multiple octaves. The convolution of the image and the Gaussian operator creates the blurred images. The second step is Laplacian of Gaussian (LoG) approximation, which is the calculation of the differences of the images in the scale for each octave. Using approximation of LoG saves computation/processing time. By picking out the maximums from the images resulting from the differences of images, it creates a set of key points. Some of these points are not useful, so the ones representing low contrast features and edges are removed from the set. Assigning each key point an orientation is the next step. This involves creating a histogram of the magnitude and orientation of the gradient. The histogram indicates the highest peaks or best gradient direction and assigns that direction to the key point. This orientation provides rotation invariance [10]. The final step is the generation of the features from the key points.

OpenCV uses Euclidean distance to match the key points and has the option to use Manhattan distance. Additional filtering can be done to create a “better” set of matches; for instance, OpenCV’s function radiusMatch would work since a pair is only kept if their distance is less than a given maximum distance. Another possible way to filter the matches is the following code:

```
vector<KeyPoint> sift_AL(Mat &src){
	SIFT siftObj = SIFT();
	Mat src_gray, mask, out;
	mask = Scalar::all(0);
	vector<KeyPoint> keypoints;
	siftObj(src, mask, keypoints);//overloaded function operator
	return keypoints;
}

// C:\OpenCV231\samples\cpp\tutorial_code\features2D\SURF_descriptor.cpp
double sift_cmp(Mat &test, Mat &train){
	//-- Step 1: Detect the keypoints using SIFT Detector
	vector<KeyPoint> testKP = or.sift_AL(test);
	vector<KeyPoint> trainKP = or.sift_AL(train);
	//-- Step 2: Calculate descriptors (feature vectors)
	SiftDescriptorExtractor extractor;
	Mat descriptors_1, descriptors_2;
	extractor.compute(test, testKP, descriptors_1);
	extractor.compute(train, trainKP, descriptors_2);
	//-- Step 3: Matching descriptor vectors with a brute force matcher
	BruteForceMatcher<L2<float>> matcher;
	std::vector<DMatch> matches;
	matcher.match(descriptors_1, descriptors_2, matches);
	if(matches.size() == 0)
		return 0;
	double max_dist = 0; double min_dist = 100;
	//-- Quick calculation of max and min distances between keypoints
	for(int i = 0; i < descriptors_1.rows; i++){
		double dist = matches[i].distance;
		if(dist < min_dist) min_dist = dist;
		if(dist > max_dist) max_dist = dist;
	}
	//-- Draw only "good" matches (i.e. whose distance is less than
2*min_dist)
	//-- radiusMatch can also be used here.
	std::vector<DMatch> good_matches;
	for(int i = 0; i < descriptors_1.rows; i++){
		if(matches[i].distance < 2*min_dist){
			good_matches.push_back(matches[i]);
		}
	}
	return good_matches.size();
}
```

2.5. Speeded Up Robust Features (SURF)

By abstracting the process of SURF, it can be described in three steps: feature detection, descriptor extraction, and matching.

The Hessian matrix is the basis for the detection of the feature points because of its good performance in computation time and accuracy [11]. Using the Hessian's determinant determines the location and scale. Part of computing Gaussian second order derivatives is using integral images and like SIFT, LoG approximations. In addition, the computation includes the use of box filters. The convolution of the Gaussian second order derivatives and the original image are the elements of the Hessian matrix. The last step of feature detection is the finding of the feature points by applying non-maximum suppression and interpolating the maxima of the determinant.

The descriptor extraction can be summed up in two steps, orientation assignment and descriptor extraction. Going into further detail, orientation assignment starts with the computation of the Haar-wavelet responses in the x and y direction, which then uses vectors to represent the responses [11]. Every slide of the window that sums the values within it creates a vector. The longest of those vectors determines the orientation. The second step of feature detection can be broken down into further steps. Centering on the interest point and oriented in the direction found in the previous step is a square region. That portion splits into smaller regions. The smaller regions apply the Haar-wavelet in the horizontal and vertical directions with respect to the orientation of the interest point. The sums of the responses for the subregions create parts of the feature vectors. The other part of the feature vector includes polarity and intensity information.

Final step is matching the key points. In OpenCV, Euclidean distance or Manhattan distance determines if a pair of key points is a match. Adding another layer of filtering to the previous one could create "better" matches. For example, the code below is one way and OpenCV offers the function radiusMatch (a pair must have a distance less than a given maximum distance).

OpenCV's implementation of SURF takes a value, hessianThreshold (minHessian in code segment below). According to [12], only features, whose Hessian is larger than the hessianThreshold are retained by the detector; therefore, the larger the value, the less keypoints you will get, so a good default value could be from 300 to 500, depending from the image contrast. Hence, depending on the image, it may be necessary to experiment with the hessianThreshold.

```
vector<KeyPoint> surf_AL(Mat &src){
	Mat out;
	int minHessian = 400;
	SurfFeatureDetector detector(minHessian);
	vector<KeyPoint> keypoints;
	detector.detect(src, keypoints);
	return keypoints;
}

// C:\OpenCV231\samples\cpp\tutorial_code\features2D\SURF_descriptor.cpp
double surf_cmp(Mat &test, Mat &train){
	//-- Step 1: Detect the keypoints using SURF Detector
	vector<KeyPoint> testKP = or.surf_AL(test);
	vector<KeyPoint> trainKP = or.surf_AL(train);
	//-- Step 2: Calculate descriptors (feature vectors)
	SurfDescriptorExtractor extractor;
	Mat descriptors_1, descriptors_2;
	extractor.compute(test, testKP, descriptors_1);
	extractor.compute(train, trainKP, descriptors_2);
	//-- Step 3: Matching descriptor vectors with a brute force matcher
	BruteForceMatcher<L2<float>> matcher;
	std::vector<DMatch> matches;
	matcher.match(descriptors_1, descriptors_2, matches);
	if(matches.size() == 0)
		return 0;
	double max_dist = 0; double min_dist = 100;
	//-- Quick calculation of max and min distances between keypoints
	for(int i = 0; i < descriptors_1.rows; i++){
		double dist = matches[i].distance;
		if(dist < min_dist) min_dist = dist;
		if(dist > max_dist) max_dist = dist;
	}
	//-- Draw only "good" matches (i.e. whose distance is less than
2*min_dist)
	//-- radiusMatch can also be used here.
	std::vector<DMatch> good_matches;
	for(int i = 0; i < descriptors_1.rows; i++){
		if(matches[i].distance < 2*min_dist){
			good_matches.push_back(matches[i]);
		}
	}
	return good_matches.size();
}
```

3. MOTIVATION

Currently, researchers use image parsing without much explanation of their selection for approach since that is not the main goal. For instance, the objective of Sudderth et al. in [13] was to describe scenes using transformed Dirichlet processes. A Dirichlet process is a distribution over distributions [14]. In other cases, researchers only focus on one area of image parsing like [3], [5], [11], and [15]. For instance, a paper may focus on improving just image segmentation or just object recognition. Accordingly, there is little focus on improving the unified image parsing process.

There is no shortage of information on image segmentation and object recognition, and others use this information to solve this specific problem rather than looking at amalgamating the two. The focus of this effort will be to look at the integration of those specific algorithms.

4. METHODS, ASSUMPTIONS, AND PROCEDURES

The training and test set of images come from the PASCAL (pattern analysis, statistical modeling, and computational learning) Visual Objects Challenge (VOC) 2005. VOC 2005's training set contains images of motorcycles (from various angles), cars (from the side and rear), bicycles, and people [16], see APPENDIX. Each test and training image has only one to three objects to recognize. Each image has an associated annotation file that contains the number of objects for that image, the correct label for each object, and the coordinates to create a bounding box around the object, see below:

```
# PASCAL Annotation Version 1.00

Image filename : "VOC2005_1/PNGImages/UIUC_TestImages/test-15.png"
Image size (X x Y x C) : 302 x 146 x 1
Database : "The VOC2005 Dataset 1 Database (UIUC)"
Objects with ground truth : 3 { "PASscarSide" "PASscarSide" "PASscarSide" }

# Note that there might be other objects in the image
# for which ground truth data has not been provided.

# Top left pixel co-ordinates : (1, 1)

# Details for object 1 (“PASscarSide”)
Original label for object 1 “PASscarSide” : “carSide”
Bounding box for object 1 “PASscarSide” (Xmin, Ymin) – (Xmax, Ymax) : (8, 59) – (107, 98)

# Details for object 2 (“PASscarSide”)
Original label for object 2 “PASscarSide” : “carSide”
Bounding box for object 2 “PASscarSide” (Xmin, Ymin) – (Xmax, Ymax) : (106, 59) – (205, 98)

# Details for object 3 (“PASscarSide”)
Original label for object 3 “PASscarSide” : “carSide”
Bounding box for object 3 “PASscarSide” (Xmin, Ymin) – (Xmax, Ymax) : (199, 60) – (298, 99)
```

Annotation file contents for test-15.png

A few of the files contain errors when it comes to the coordinates of the bounding box and the process of reading in the annotation files corrects this. It checks that the minimum is less than the maximum and that the coordinates given do not go outside the bounds of the image. To reduce processing time, a subset of the images are used; 50 motorcycle images from ETHZ_motorbike-testset folder (represented as motorbike-test in Table 3) and 50 car images from UIUC_TestImages folder (labeled test in Table 4) were used as the test set and 100 images of each object type was used as the training set. The training set consists of images from the folders Caltech_cars, Caltech_motorbikes_side, ETHZ_sideviews-cars, TUGraz_bike, TUGraz_cars, and TUGraz_person.

Table 3. PASCAL VOC Dataset 2005 Statistics, Dataset 1

	motorbike-test		**test**		**Training**	
	Images	**Objects**	**Images**	**Objects**	**Images**	**Objects**
Testing	95	102	136	164	-	-
Training	-	-	-	-	1066	1424
Select set	50	56	50	66	200	253

Using OpenCV 2.3.1 and C++, we take advantage of the built-in functionality and have a better processing time as compared to Java. To segment images we used OpenCV's Sobel, Canny, and Mean-Shift Segmentation implementations. Additionally, we used OpenCV's object recognition implementations of SIFT and SURF, which are the original patented versions. OpenCV's implementations of the methods require a few function calls, thus avoiding having to reimplement them.

Even after making these decisions, the processing of the images is still lengthy. However, this is expected. The program reads in the image as well as an annotation file for each training image. This information produces the training segments. Throughout the rest of the report, reference images, reference objects, or reference segments refers to training images, training objects, or training segments respectively.

The test set has a similar process applied (Code Segment 1). An image and its annotation file are read into the program to get the bounding box and the correct label for each object within the image. Each segmentation algorithm segments the test image. The segment used as the test segment is the one that most overlaps the bounding box around the actual object. Using a combination of the minimum x and y coordinates and the maximum x and y coordinates of the segment converts the segment into a box. The box-segment that most overlaps the object's true bounding box produced from the annotation file is the one used.

```
imgFilepath = File path and name of image
annotFile = Get annotation file by using imgFilepath and replacing
"PNGImages" with "Annotations"
numObjects = Parse annotFile to get number of objects in file
For each object in annotFile
    objLabel = Parse out object's label
    bounds[] = Parse out coordinates of object's bounding box (min x and
y, max x and y)
    segment = Create a segment from bounds (only do if creating training
data)
    Create a data structure that holds label and bounds
```

Code Segment 1. Process for Creation of Test and Training Segments

For every test segment there is a comparison to every reference segment. For each combination of image segmentation algorithms and object recognition algorithms a comparison is made. After the comparison, it tracks the label and how much the reference segment and test segment have in common.

The object recognition approach used determines the method for determining how much the reference and test segments have in common. Edge map approaches are naively calculated. Starting at the top left of the segments as the origin and using smallest number of columns and rows between the two segments is the area checked for common pixel colors. Therefore, for each pixel within that area, if the intensity value of the pixel in the same position in the test segment as the one in the reference segment is the same, then that is considered a match. The number of matching pixels divided by the total number of pixels in the area determines the similarity of the segments.

$$Commonality\ for\ edge\ detection\ methods = \frac{number\ matching\ pixels\ (based\ on\ their\ intensity)}{total\ number\ of\ pixels\ in\ the\ area\ being\ checked} \quad (2)$$

However, if SIFT or SURF is used for object recognition, then the measure of similarity is the number of matches. A brute force matcher (or BruteForceMatcher in OpenCV) finds matches based on Euclidean distance [12] and for better results, only the matches that are less than two times the smallest distance among matches are kept.

$$\text{d} = \text{Euclidean distance between a pair of key points}$$
$$\text{Commonality for SIFT and SURF} = \text{d} < 2 * \text{minimum distance (of the matches)} \quad (3)$$

Since we aware that handling factors like rotation and scale are important to deal with when using a real data set, we included Scale Invariant Feature Transform (SIFT) ([15], [17]) and Speeded Up Robust Features (SURF) ([11], [18]) for object recognition options. They are exceptional in that they can handle images of different scales and rotations [19].

5. RESULTS AND DISCUSSION

It is wasted effort when an analyst or warfighter should have his/her focus on finding new information, things not so easily discovered. As the Air Force Research Laboratory’s goal is to help the analyst and warfighter, this effort only uses segmentation algorithms that do not require user input as an analyst and warfighter have much data to go through. However, the segmentation methods implemented tended to over-segment. The average number of segments in the test object’s bounding box for Canny edge detection is 47, 2075 for mean-shift, and 538 for Sobel edge detection.

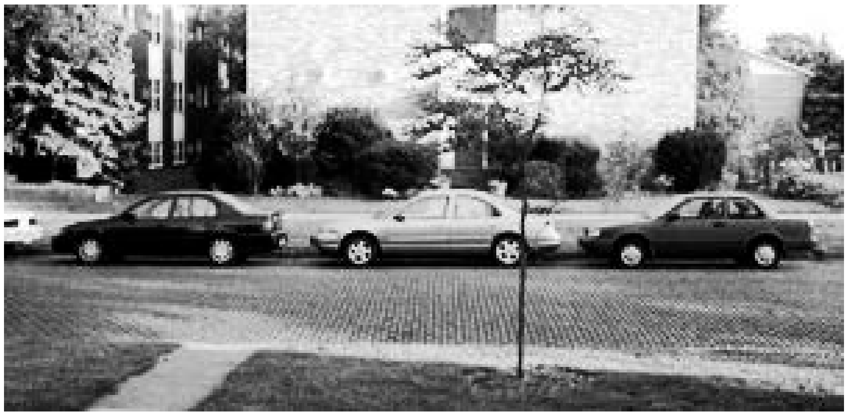
Figure 1. Original image, test-15.png in test set [16]

Looking at Figure 1, Figure 2, Figure 3, and Figure 4, you can see the segmentation results for test image test-15.png where each color represents a segment; according to human vision, we see they have imperfect segmentation. Consequently, the identification of the object becomes more difficult. If the test segment is too small (includes a part of the object) or too large (includes the object and one or more objects), then the measure of similarity will be different if the test segment was just the object. Therefore, an imperfect test segment could lead to an incorrect labeling.

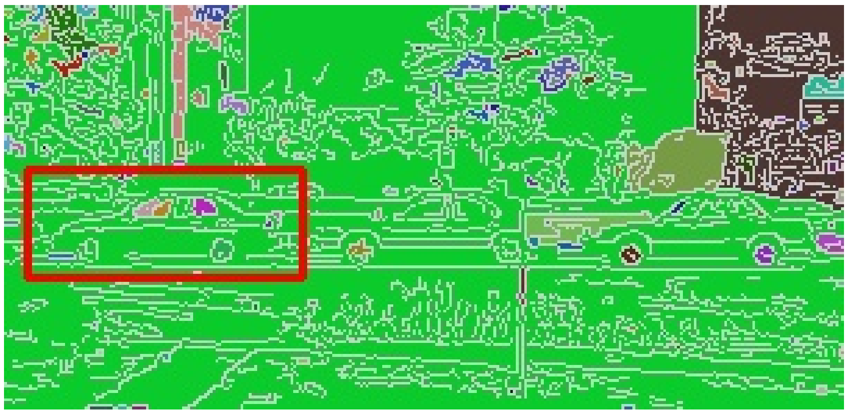
Figure 2. Canny edge detection applied to test-15.png

In Figure 2, it appears that the majority of the image is one segment. Applying a Gaussian blur smoothes the image as well as removes noise. Therefore, texture such as the image's ground, does not seem to be as big of an issue as for the other methods. As for the imperfect segmentation, the edges may not have been thick enough or complete enough. The algorithm itself creates thin edges. This is because of two of the three criteria for edge detector performance, localized edge points (points marked as edges by detector and the center of a true edge should have a minimum distance between them) and single edge point response (the detector should not identify multiple edge pixels where only a single edge point exists) [2]. Nevertheless, this may exacerbate the issue because of the thresholds used for the hysteresis. The thresholds are parameters for the method used by OpenCV. One set of thresholds may work for one image, but not another. One way to thicken edges is to use mathematical morphology ([20]), specifically the dilation operation.

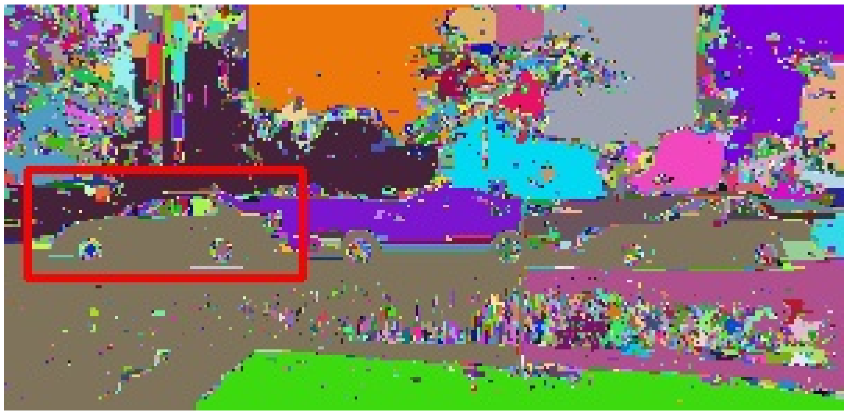

Figure 3. Mean-shift applied to test-15.png

Figure 3 shows the mean-shift results. Once again, the segmentation is imperfect; a possible reason for the flawed segmentation is the parameters given to the OpenCV functions. Like Canny, the variables to the mean-shift functions could work well for one image and not another.

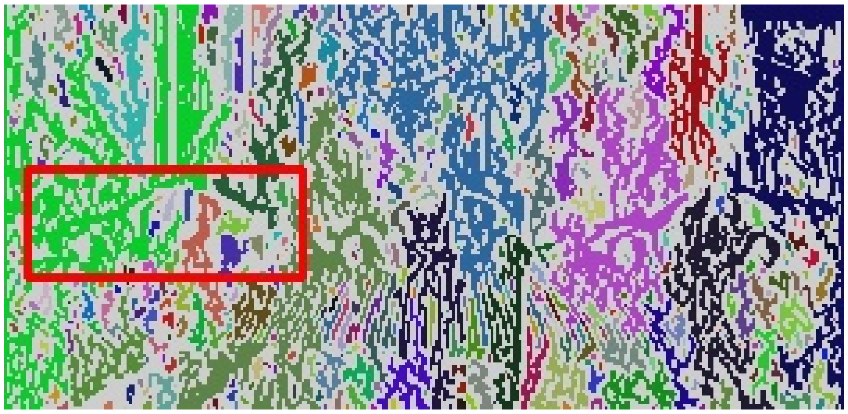

Figure 4. Sobel edge detection applied to test-15.png

Figure 4 is the output for Sobel edge detection. An advantage of Sobel is the twos in the filters, which provides some smoothing; however, the image could use additional smoothing before applying the operators. In addition to the smoothing or instead of, applying a threshold to the gradient image (the product of Sobel edge detection) is beneficial. Using both methods would reduce the textured look of the image by keeping the most prominent edges.

If the segments are imperfect, this could lead to error in the object recognition process. The metrics used are:

$$Precision = \frac{relObjs \cap retObjs}{retObjs} \tag{4}$$

$$Recall = \frac{relObjs \cap retObjs}{relObjs} \tag{5}$$

$$F - measure = 2 * \frac{precision * recall}{precision + recall} \tag{6}$$

In equations (4) and (5), relObjs is the number of objects with the type that is the current focus. For instance, if the test set has 30 motorbike objects, then relObjs is 30. retDocs is the number of objects assigned the type being looked at. For example, if of all the objects in the test set 20 of them received the label bike, then retDocs for bike would be 20. Looking at Table 3, it shows that the performance is poor for the most part.

Table 4. Precision, Recall, and F-measure for All Combinations Using Loose Labeling

	Canny				Mean Shift				Sobel			
	Sobel	**Canny**	**SIFT**	**SURF**	**Sobel**	**Canny**	**SIFT**	**SURF**	**Sobel**	**Canny**	**SIFT**	**SURF**
	Object = motorbike											
Precision	0.35	0.00	0.44	0.15	0.35	0.04	0.38	0.05	0.45	0.79	0.44	0.41
Recall	0.52	0.00	0.57	0.04	0.54	0.02	0.55	0.04	0.96	0.55	0.88	0.82
F-measure	0.42	0.00	0.50	0.06	0.42	0.02	0.45	0.04	0.61	0.65	0.59	0.55
	Object = car											
Precision	0.33	0.67	0.00	1.00	0.29	1.00	1.00	0.50	0.00	0.88	1.00	0.00
Recall	0.20	0.06	0.00	0.05	0.15	0.11	0.02	0.02	0.00	0.32	0.02	0.00
F-measure	0.25	0.11	0.00	0.09	0.20	0.19	0.03	0.03	0.00	0.47	0.03	0.00

Additionally, by observation of the results in Table 3 (where green is the best performance for the measure), no combination has the best performance for all three metrics. Conversely, F-measure indicates that the combination of Sobel for image segmentation and Canny for recognition is best.

Table 4 is the "overall performance" of each combination, which means the number of correctly labeled test segments divided by the total number of test segments. Based on this metric, using Sobel for segmentation and Sobel for object recognition has the best performance. Despite the combination having the best performance, labels of less than a half of the test segments were correct.

Table 5. Performance

Performance – Strict (%)

	Canny	**Mean Shift**	**Sobel**
Sobel	23.77	24.59	44.26
Canny	0.00	1.64	29.51
SIFT	26.23	25.41	40.98
SURF	3.28	2.46	37.70

Performance – Loose (%)

	Canny	**Mean Shift**	**Sobel**
Sobel	34.43	32.79	44.26
Canny	3.28	6.56	42.62
SIFT	26.23	26.23	40.98
SURF	4.10	2.46	37.70

Strict means that the view as well as the object type had to be correct. Whereas with loose, only the type of the object had to be correct.

6. CONCLUSION

Looking at the numbers produced by the metrics in Table 3, we can see that no one combination out does the others for all three metrics. Additionally, according to Table 3 and F-measure (which considers both precision and recall), we should use Sobel for segmentation and Canny for recognition.

If we only consider the number of test segments correctly labeled divided by the total number of test segments, then that clearly indicates the combination of Sobel and Sobel. Despite being the clear winner in that case, the combination only performed so well because of the number of correctly labeled motorbikes. Looking at Table 3, we can see that using SIFT and SURF on cars did not work well. Additionally, the performance was not stellar according to Table 4; labels of less than half of the test segments were correct. Such results would not be useful to an analyst or warfighter.

Despite the performance of the combinations used in this effort, there are plenty of other algorithms to try. As mentioned previously, there are multiple categories within image segmentation and object recognition and each of those categories have one or more algorithms.

7. FUTURE WORK

Through research for this effort, there are two key things to focus on for improvement, implementations and data sets.

Most of OpenCV's implementations of the methods require parameters. These parameters could be partially hindering performance. Still, tweaking parameters only gets us so far and finding well performing parameters generic enough to work on all possible cases is near if not impossible. Another opportunity for future work is to implement other segmentation and object recognition algorithms. Since the input for the object recognition methods comes from segmentation, we would look at segmentation algorithms first. Furthermore, we could explore machine learning approaches like k-nearest neighbor or neural networks to possibly improve precision, recall, and F-measure.

Additionally, the best way of choosing which label associated with a reference image is best for the test segment could use some exploration since there is a multitude of ways to do this. For instance, when using SIFT and SURF, Manhattan distance could be used instead of Euclidean to determine matching key points. Another thing to try is to use a distance formula to decide if pixel values match when using edge detection methods. An additional approach of choosing a label could be applying checks for rotation and scaling to the edge detection methods.

The results of this effort indicate that the combination to use is Sobel for image segmentation and Canny or Sobel edge detection for object recognition. To further support this result, a more varied data set would be useful since the data set used for this effort only had two types for the test segments, motorbikes and cars. Another variation to add to the test set is images with rotated objects. An additional problem is occlusion (and includes its own set of algorithms); addressing this problem is important because not all objects will be unobstructed in the images. The test set does contain images with occlusion; however, the segmentation and recognition algorithms implemented do not directly deal with this issue.

Finally, a potential experiment to improve performance is to use video or sequence of images. By taking this approach, we could then get an average over several images or frames to determine an object's label.

8. BIBLIOGRAPHY

[1] J. Prewitt, "Object enhancement and extraction," *Picture Processing and Psychopictorics,* vol. 75, 1970.

[2] R. Gonzalez and R. Woods, Digital Image Processing, Upper Saddle River: Pearson Prentice Hall, 2008.

[3] I. Sobel, "Camera models and machine perception," Stanford University, Palo Alto, 1970.

[4] D. Marr and E. Hildreth, "Theory of edge detection," *Proceedings of the Royal Society of London,* vol. 207, pp. 187-217, 1980.

[5] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vols. PAMI-8, no. 6, pp. 679-698, 1986.

[6] "Segmentation (image processing)," Wikipedia, [Online]. Available: http://en.wikipedia.org/wiki/Segmentation_(image_processing).. [Accessed 16 October 2012].

[7] M. Treiber, An Introduction to Object Recognition: Selected Algorithms for a Wide Variety of Applications (Advances in Computer Vision and Pattern Recognition), Springer, 2010, p. 6.

[8] "Hysteresis," Merriam-Webster, Inc., [Online]. Available: http://www.merriam-webster.com/dictionary/hysteresis. [Accessed November 2012].

[9] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 24, pp. 603-619, 2002.

[10] U. Sinha, "SIFT: Scale invariant feature transform," AI Shack, 14 May 2010. [Online]. Available: www.aishack.in/2010/05/sift-scale-invariant-feature-transform/. [Accessed October 2012].

[11] H. Bay, T. Tuytelaars and L. Gool, "SURF: Speeded up robust features," *Computer Vision - ECCV 2006,* vol. 3951, pp. 404-417, 2006.

[12] *The OpenCV Reference Manual Release 2.3,* 2011.

[13] E. Sudderth, A. Torralba, W. Freeman and A. Willsky, "Describing visual scenes using transformed Dirichlet processes," *Advances in Newural Information Processing Systems 18,* pp. 1299-1306, 2005.

[14] Y. Teh, "Dirichlet process," in *Encyclopedia of Machine Learning*, Springer, 2010, pp. 280-287.

[15] D. Lowe, "Object recognition from local scale-invariant features," *The Proceedings of the Seventh IEEE International Conference on Computer Vision,* vol. 2, pp. 1150-1157, 1999.

[16] M. Everingham, A. Zisserman, C. Williams, L. Gool and et al., "The 2005 PASCAL Visual Object Classes Challenge," [Online]. Available: http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2005/. [Accessed 2012].

[17] D. Lowe, "Method and apparatus for identifying scale invariant features in an image and use of same for locating an object in an image". U.S. Patent 6711293, 23 March 2004.

[18] R. Funayama, H. Yanagihara, L. Gool, T. Tuytelaars, H. Bay and et al., "Robust interest point detector and descriptor". U.S. Patent 2009238460, 24 September 2009.

[19] "Scale-invariant feature transform," Wikipedia, [Online]. Available: http://en.wikipedia.org/wiki/Scale-invariant_feature_transform. [Accessed 16 October 2012].

[20] R. Haralick, S. Sternberg and X. Zhuang, "Image analysis using mathematical morphology," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vols. PAMI-9, no. 4, pp. 532-550, 1987.

[21] "The PASCAL Visual Object Classes Homepage," [Online]. Available: http://pascallin.ecs.soton.ac.uk/challenges/VOC/. [Accessed 5 March 2013].

APPENDIX

Below is a subset of images from the data set for visual reference. All images in the appendix come from [16]. According to [21], most images in the data set are from existing public data sets.

Figure 5. motorbikes002.png From Test Set

Figure 6. motorbikes044-rt.png From Test Set

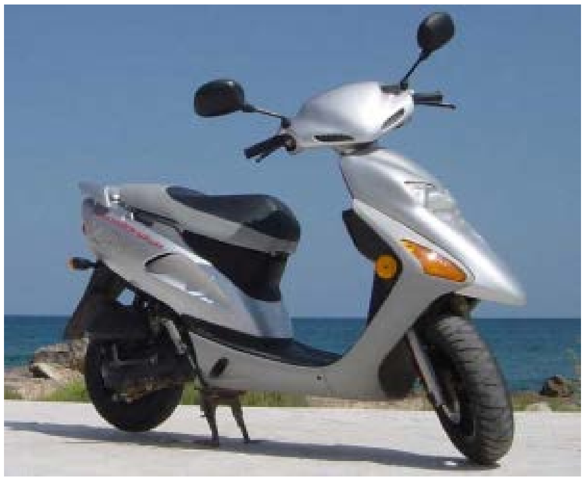

Figure 7. motorbikes048-rt.png From Test Set

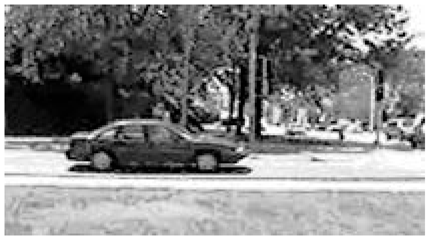

Figure 8. test-0.png From Test Set

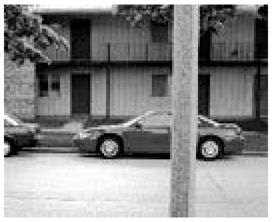

Figure 9. test-31.png From Test Set

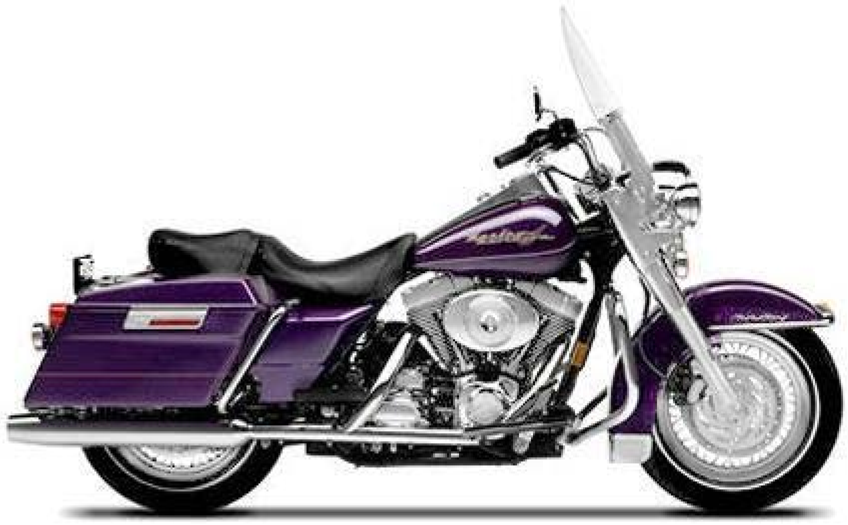

Figure 10. 0002.png From Training Set

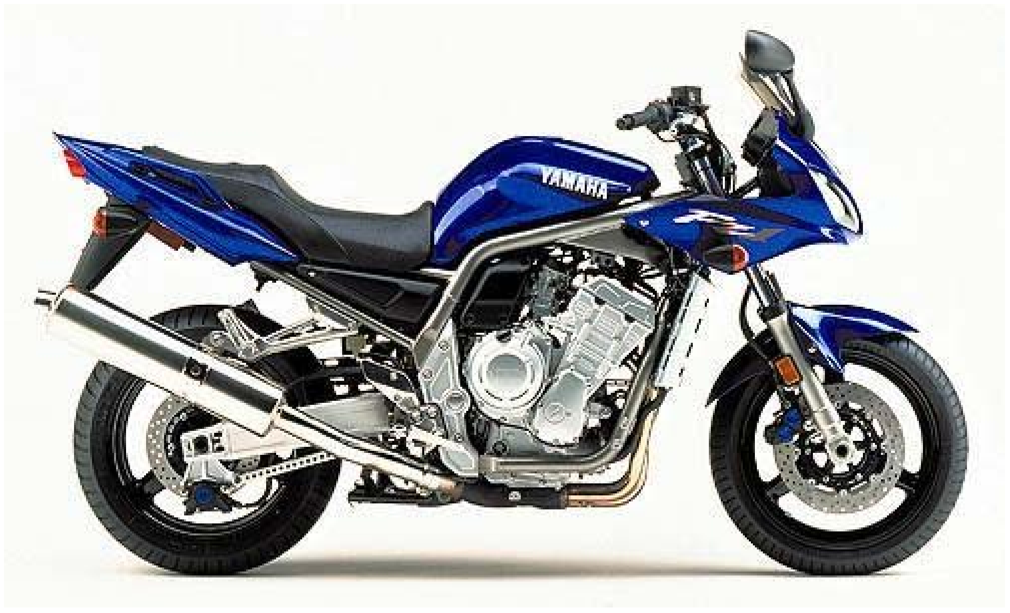

Figure 11. 0025.png From Training Set

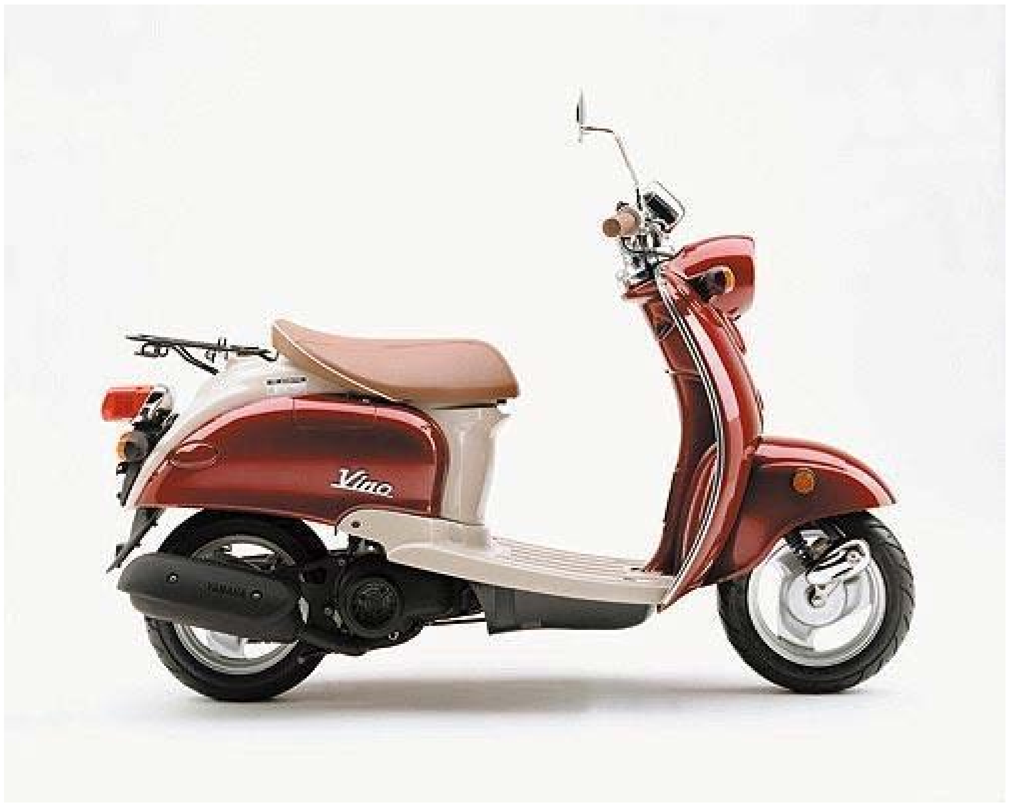

Figure 12. 0046.png From Training Set

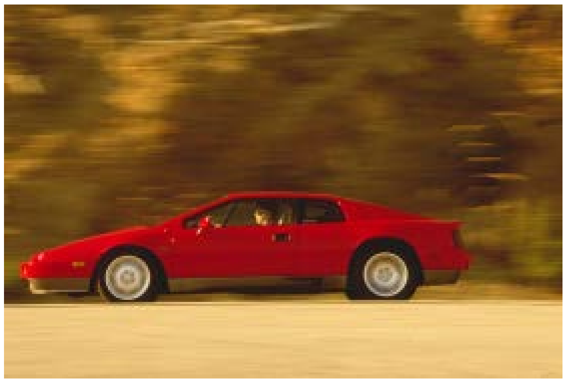

Figure 13. 29091-sml.png From Training Set

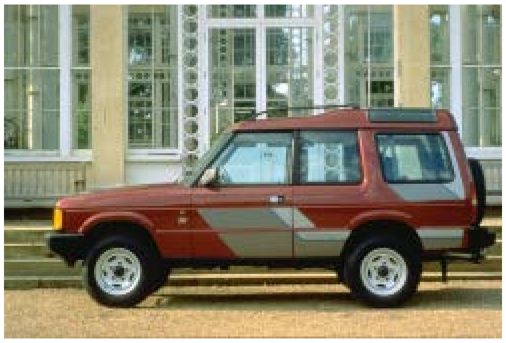

Figure 14. 354002-sml.png From Training Set

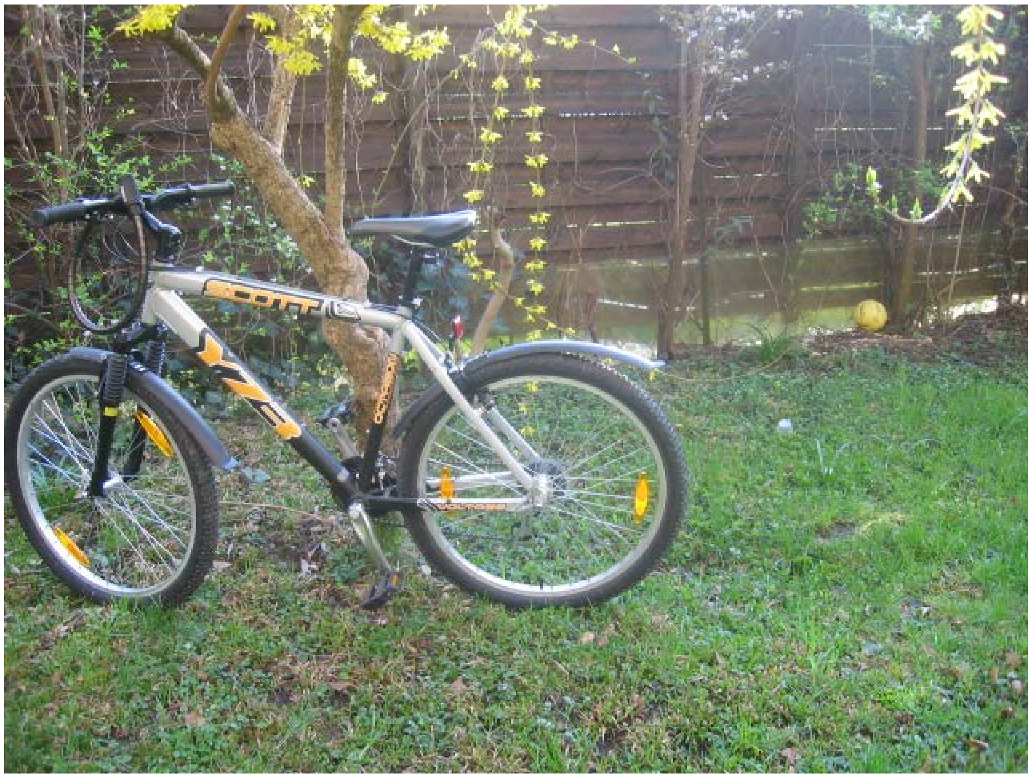

Figure 15. bike_002.png From Training Set

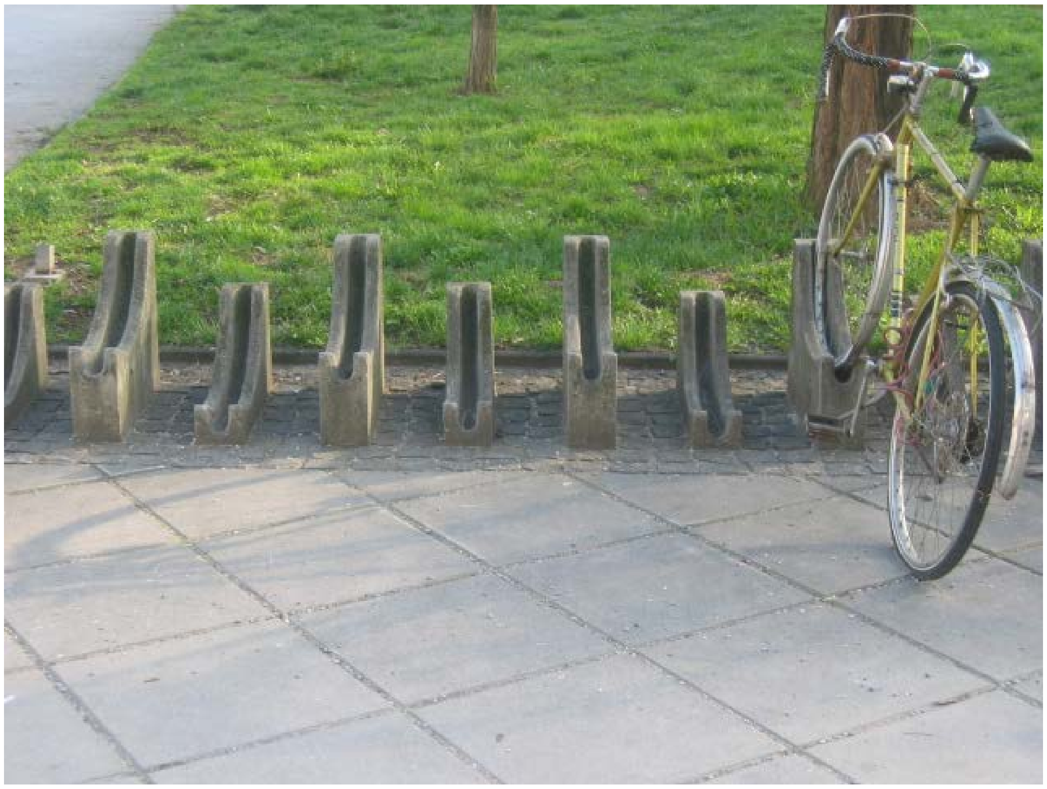

Figure 16. bike_005.png From Training Set

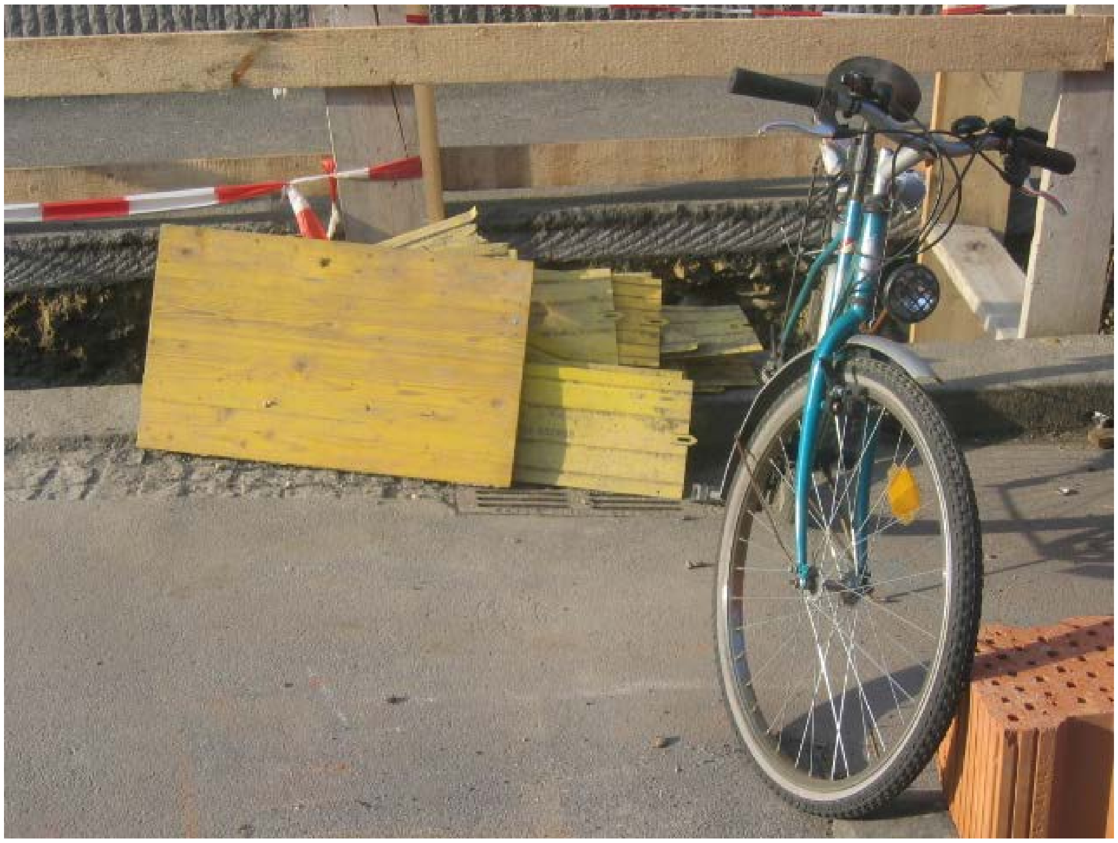

Figure 17. bike_008.png From Training Set

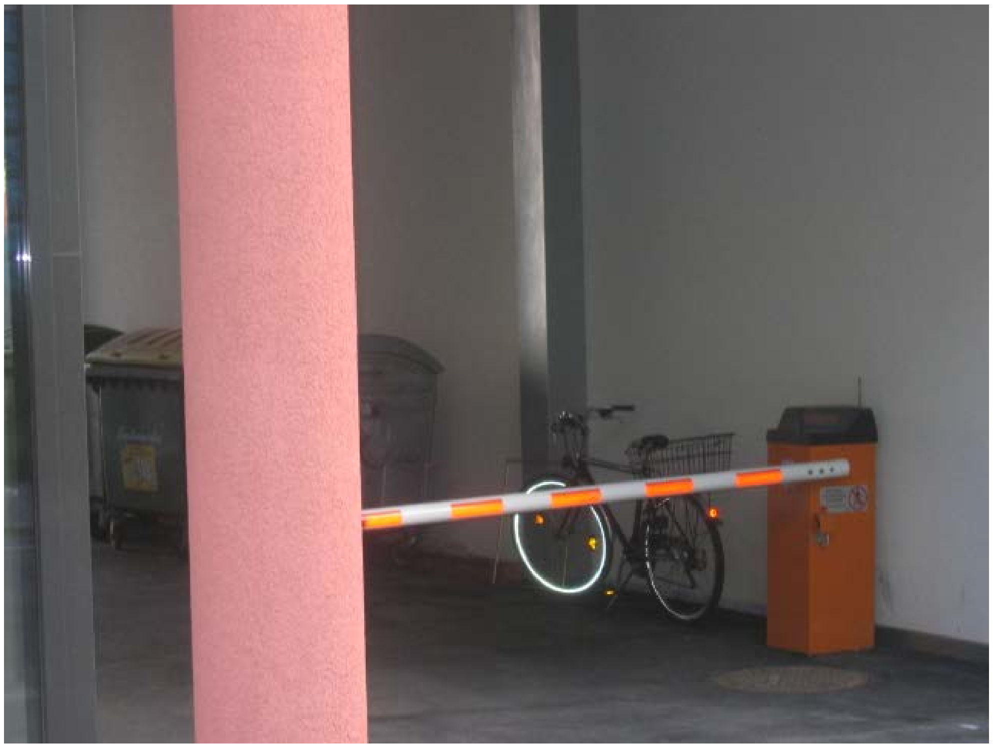

Figure 18. bike_014.png From Training Set

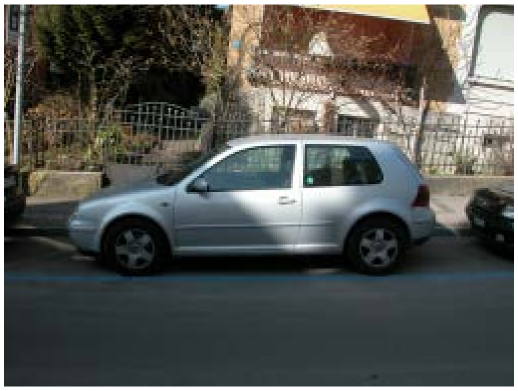

Figure 19. car-pic40-sml.png From Training Set

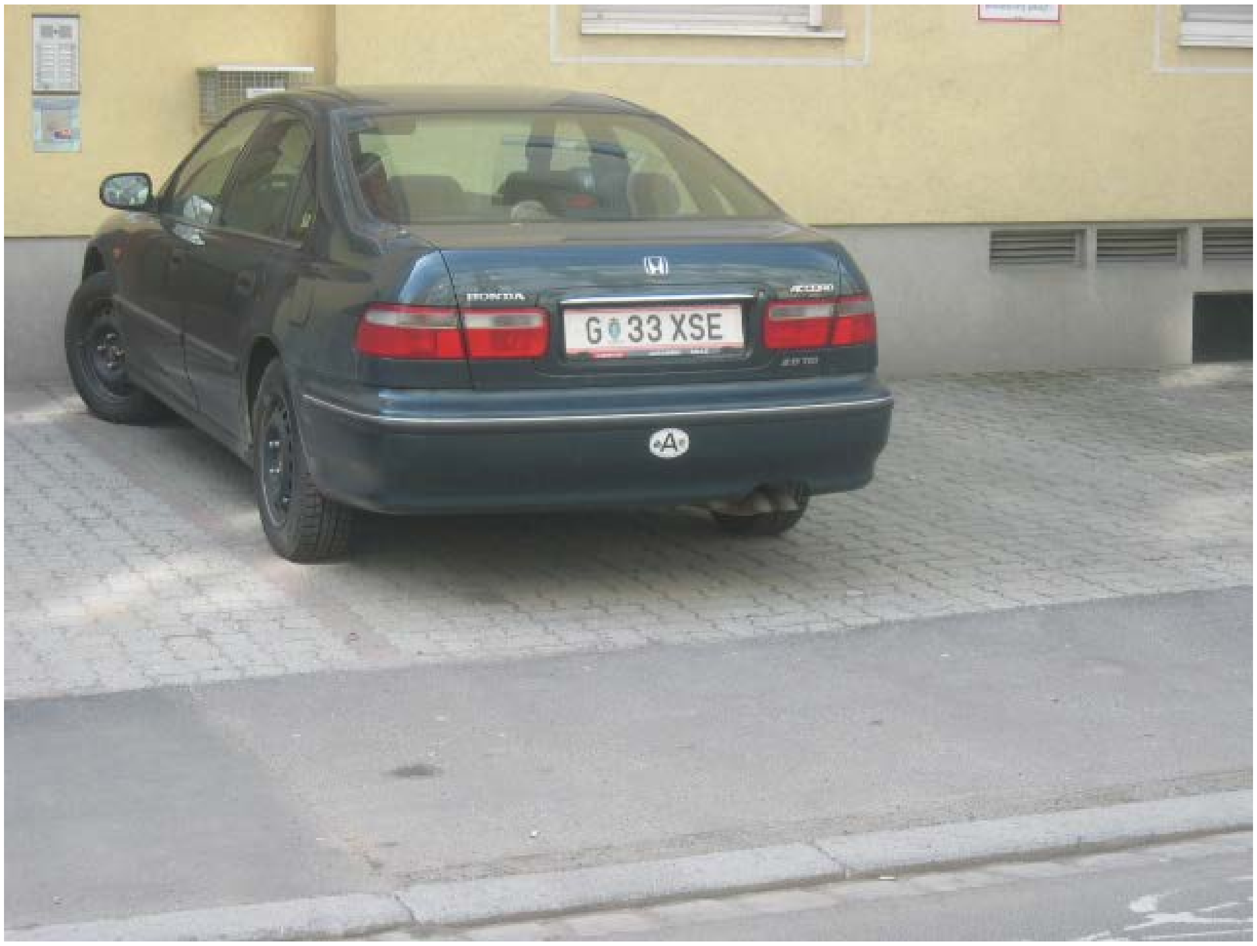

Figure 20. carsgraz_001.png From Training Set

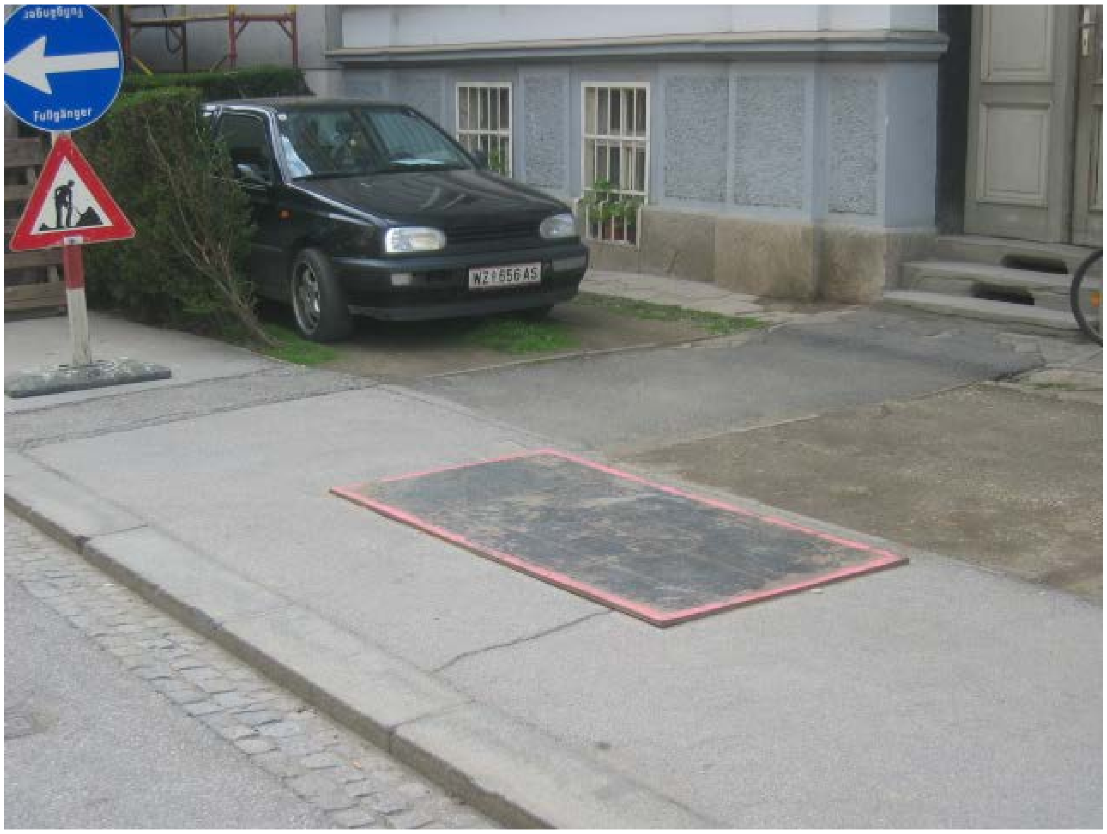

Figure 21. carsgraz_006.png From Training Set

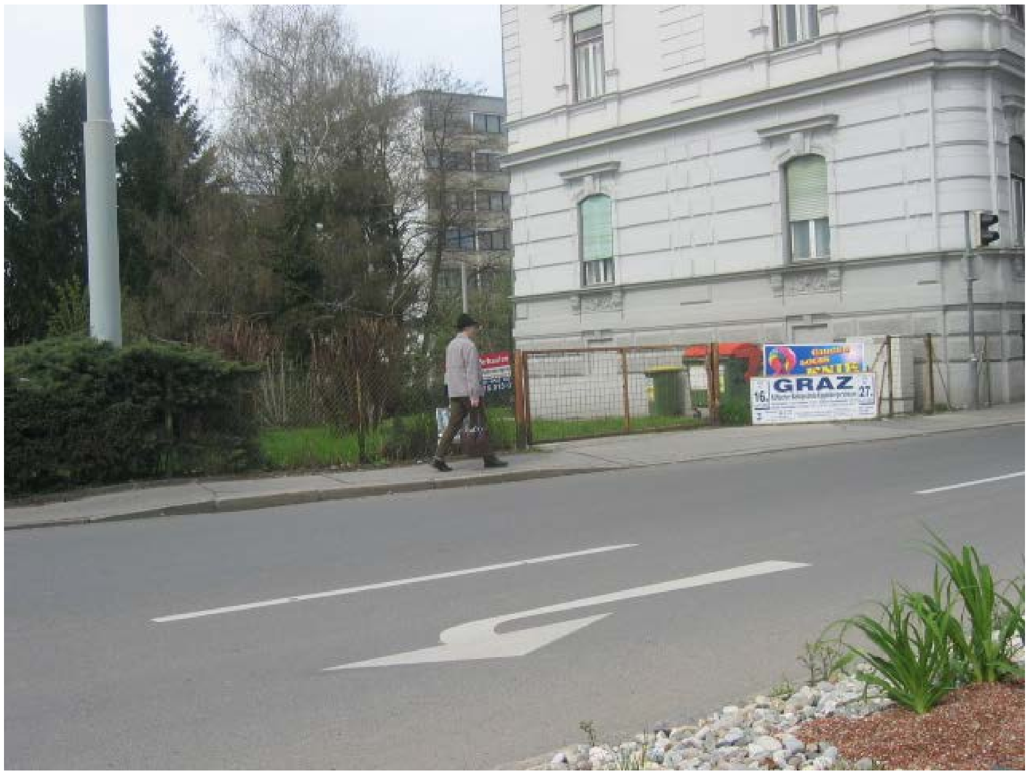

Figure 22. person_002.png From Training Set

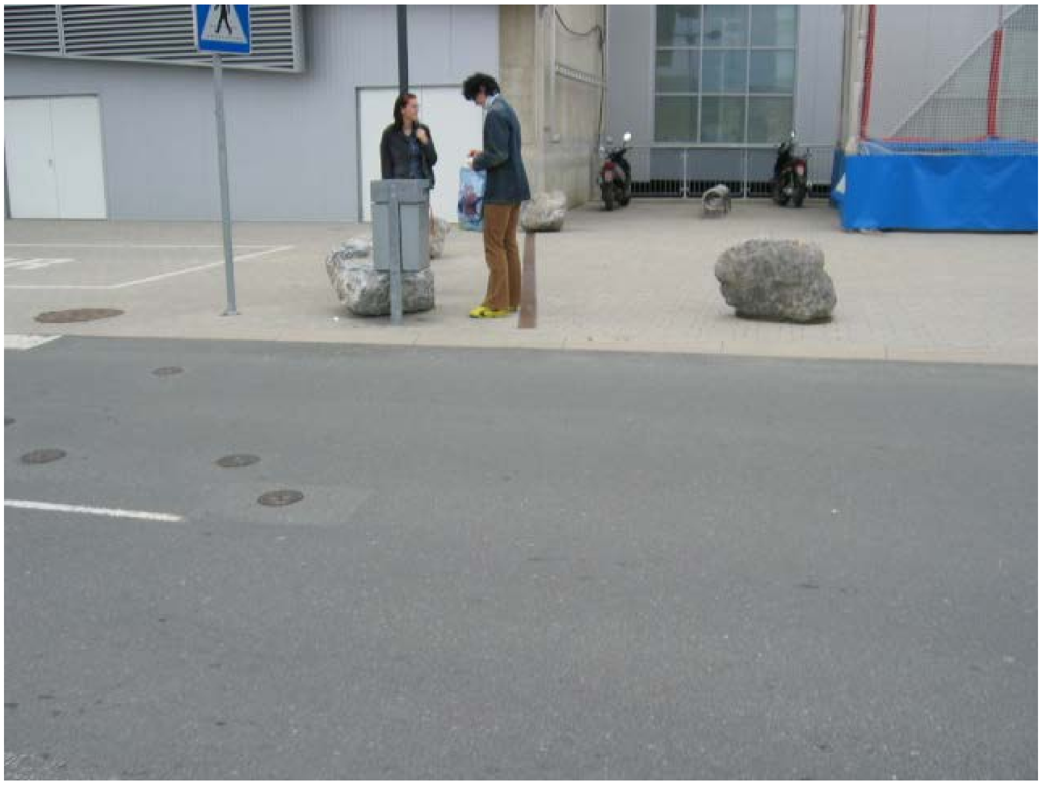

Figure 23. person_005.png From Training Set

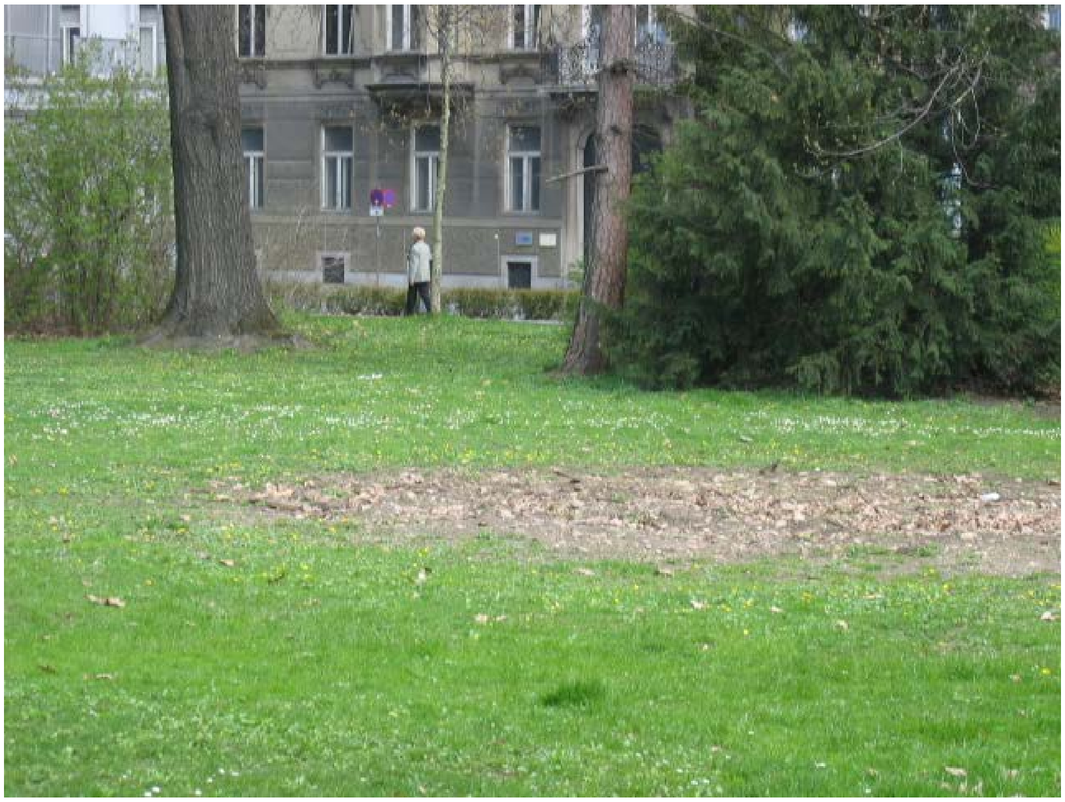

Figure 24. person_020.png From Training Set

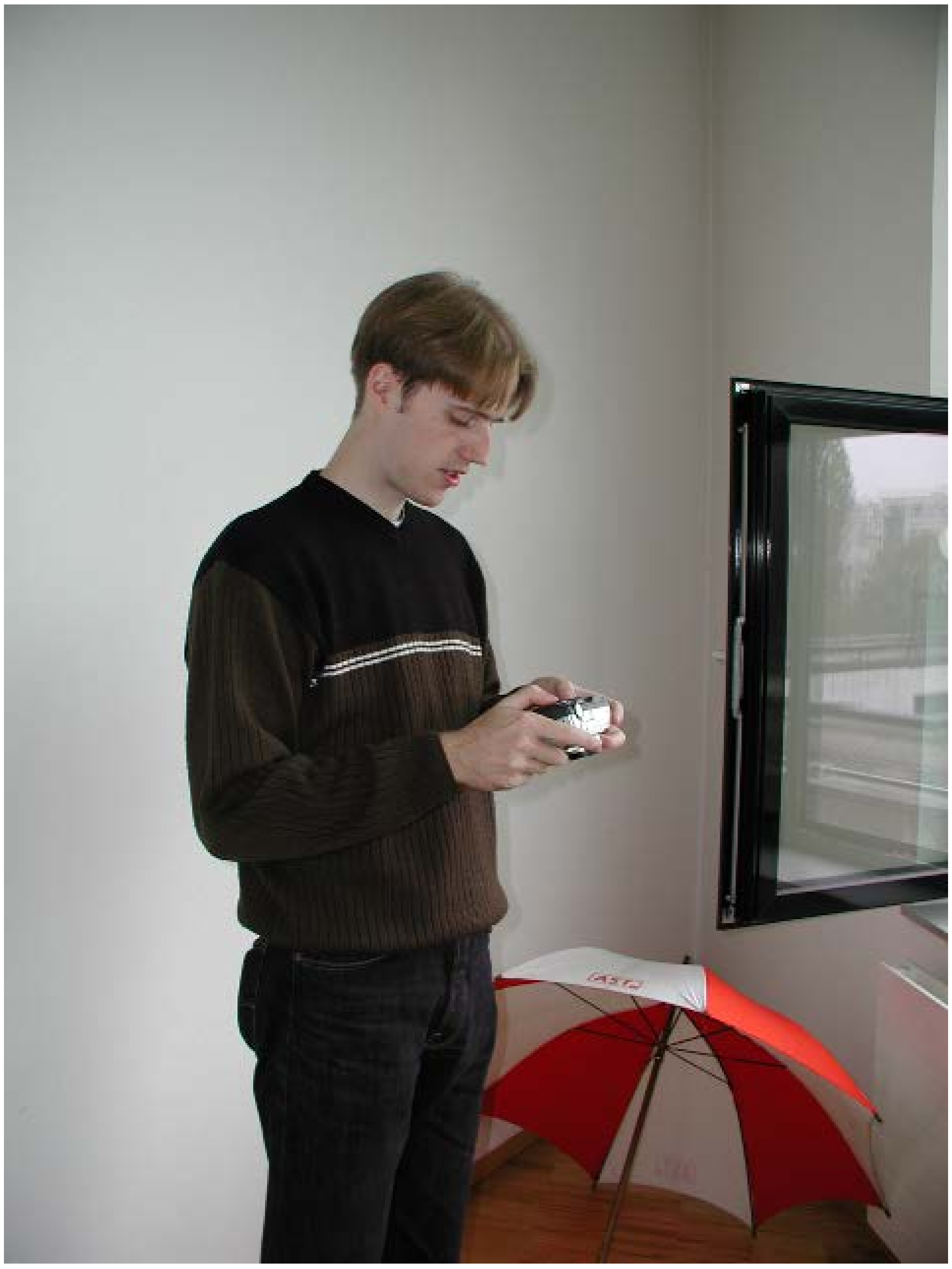

Figure 25. person_048.png From Training Set

LIST OF ACRONYMS

LoG	Laplacian of Gaussian
PASCAL	Pattern analysis, statistical modeling, and computational learning
SIFT	Scale Invariant Feature Transform
SURF	Speeded Up Robust Features
VOC	Visual Objects Challenge